PROGRAMMING FOR DATA ANALYTICS 2

Task 2

Kieran Glezer-Jones
ST10041889
October 2022

Table of Contents

1 Introduction

Fraud is a big problem in our current money transferring system. There are many different ways to perform it but, in this report, we will specifically be looking at instances where someone has flagged a transaction as fraudulent.

This report revolves around a dataset which contains millions of records where some have been reported as fraud. Using this dataset, a model will be constructed to accurately predict fraudulent behaviour. This could result in an extremely strong competitive advantage for any bank utilizing this technique.

(Carleton, 2022) explains that if someone hacks into a person's account and steals money from them, the banks are often held liable for the loses, this however, still doesn't display the bank in a great light. The banks are supposed to protect the user against fraudulent transactions, even if the user gave their password away willingly.

It might seem unreasonable to hold a bank accountable for this but when someone says my Standard Bank account was hacked, they aren't going to tell their friends that they gave their password willingly. They are going to blame the organization to save face. Alternatively, there are many, very difficult to detect, ways to gain access to someone's account, such as a chip planted in an ATM card reader. The fact of the matter is that when someone is hacked, that banks image gets damaged, and people will spread this information and the more it spreads the worse the bank looks.

So, if X bank utilizes an algorithm that can recognize fraudulent transactions, and Y bank does not. Not only does X bank look good, but Y bank starts to look bad. This is the competitive advantage that this detection algorithm provides.

With the algorithm in play, transactions can be analysed and if the algorithm thinks it looks a little strange, it could potentially hold the transaction all together and attempt to contact the person associated with the account.

2 Background

The following section covers information that is known before the start of the analysis. It goes over what is known about the dataset as well as basic information deemed necessary to understanding the report.

2.1 The Dataset

The dataset in question was sourced off of Kaggle.com and uploaded under the title of Fraudulent Transaction Prediction (Siramdasu, 2022). A link to the dataset is provided in a README file contained in the project folder.

The dataset contains over 6 million records of transactions, where only about 8 thousand of which are fraudulent. This means that over 99 percent of the data is legit.

There are over 10 features in the dataset but only 7 of them were deemed necessary to completing the model.

Variable Name	Description	Qual/Quan	Type
Type	Type of payment method used.	Qualitative	Nominal
Amount	The amount of money that was transferred	Quantitative	Continuous
oldbalanceOrig	The original balance of the person who gave the money.	Quantitative	Continuous
newbalanceOrig	The new balance of the person who gave the money.	Quantitative	Continuous
oldbalanceDest	The old balance of the person receiving the money.	Quantitative	Continuous
newbalanceDest	The new balance of the person receiving the money.	Quantitative	Continuous
isFraud	A one hot variable describing if the record is fraud (1) or legit (0)	Qualitative	Nominal

Within the type variable there are five types of payments.

Transfer Type	Description
Payment	A straight payment from a customer to a merchant
Transfer	A transfer from account to account.
Cash Out	Withdrawn money from an account
Cash In	Money put into an account
Debit	A debit card payment

2.2 Correlation

The correlation between data features will describe how strong the relationship is between them. 1 is a perfect positive relationship and -1 is a perfect negative relationship (Akoglu, 2018). Assuming that there is a perfect positive relationship, if the independent variable goes up by 10 units, the dependent variable will also go up by 10 units. If there is a perfect negative relationship, then as the independent variable increases the dependent variable will decrease.

Correlation Coefficient		Dancey & Reidy (Psychology)	Quinnipiac University (Politics)	Chan YH (Medicine)
+1	−1	Perfect	Perfect	Perfect
+0.9	−0.9	Strong	Very Strong	Very Strong
+0.8	−0.8	Strong	Very Strong	Very Strong
+0.7	−0.7	Strong	Very Strong	Moderate
+0.6	−0.6	Moderate	Strong	Moderate
+0.5	−0.5	Moderate	Strong	Fair
+0.4	−0.4	Moderate	Strong	Fair
+0.3	−0.3	Weak	Moderate	Fair
+0.2	−0.2	Weak	Weak	Poor
+0.1	−0.1	Weak	Negligible	Poor
0	0	Zero	None	None

Figure 1 correlation by (Akoglu, 2018)

2.3 Classification

Classification is a type of machine learning that takes independent X variables as input and tries to predict a dependent Y variable. The difference between regression and classification is that classification outputs a categorical variable while regression outputs a continuous variable.

To perform supervised classification, you need to have input variables and known outputs to train the model. Once the model is trained the target can be predicted afterwards. The dataset we are using has 7 useful variables, 6 of which can be used as valid inputs for the model, while the 7th variable can be used as the target. The target being the [isFraud] feature.

2.4 Descriptive Analysis

Descriptive analytics involved taking raw historical data and then data wrangling it to discover trends, patterns, and relationships (The Independent Institute of Education, 2022). This process breaking the raw data down and comparing, often visually, to other elements within the dataset with the intent of discovering how they interact with each other. This type of analysis answers the question of “what happened”.

Descriptive analysis will be performed on the dataset in question to try and gain insight into what is typical of legitimate transactions and how do fraudulent transactions differ from them.

2.5 Predictive Analysis

Predictive analysis attempts to answer the question of “what is going to happen” by taking in large amounts of historical data, discovering the patterns and relationships hidden within, and coming to a prediction based off of new data (Cote, 2021).

To perform this analysis, we will build a model that will take large amounts of historical fraud data and figure out what is typical behaviour within the input variables for both legit and fraudulent data.

2.6 Sampling

Random sampling is when we take a random small amount of data from a much larger population of data (Olken & Rotem, 1995).

The dataset in question has over 6 million records, for machine learning this is quite a large number of records, and to test a model of such capacity would take far too long for a report of this nature. So, to combat this, samples will be taken from the dataset to analyse. It’s important to note that all the fraudulent records will be taken, this is because there are already too few fraudulent records to begin with.

Depending on how consistent the trends are within the data you might need as few as 10 records. But if the trends aren't as obvious then more records might be required. The sample size will be increased and decreased depending on the model's performance during model selection and construction.

2.7 Data Wrangling

Data wrangling is a six-step process which aims to get the data ready for the model to process. The steps involve discovering, structuring, cleaning, enriching, validating and then publishing the data (Stefanski, 2022).

2.7.1 Discovering the Data

This is the part where we get comfortable with the data. We want to collect the data from its various sources and visualise it using various methods. For this dataset specifically we will look to see if we spot any patterns between transaction types and the amounts in the various balances.

2.7.2 Structuring

When we receive the data there will be a lot of columns we do not need, we will get rid of all the features which we deem unnecessary for the model to work and keep the ones we deem necessary.

2.7.3 Cleaning

This is where we make sure that the data is in the correct format with no errors such as null values.

2.7.4 Data Enriching

It is already known that there is a lack of fraudulent records in the dataset. To combat this, we will be utilizing SMOTE to generate synthetic records. This is an example of how we will enrich the data.

2.7.5 Validation

This is the step where we make sure that all the features are correct and in the correct format. All the numeric variables should be scaled, the categoric variables should be dummied, and the shape should be verified.

2.7.6 Publishing

Once the data has been verified, we can look to publish it. This is the step where we save the data for the model's use and then pass it to the model to train and test it.

3 Usability

The ultimate goal is to create a classification model that can take input and then get a prediction on whether it was a fraudulent transaction or not. The prediction will output either a 0 or a 1 and to get to that point a few requirements need to be met.

3.1 Legitimate Source

If the model is trained on numbers that hold creator bias or are randomly generated, then its performance will be jeopardized. The accuracy might be very good but its performance in the real world might not reflect that. This is because relationships in the real world are hard to generate and sometimes humans fail to pick up on them.

The provided dataset is indeed simulated. This means that it was generated via simulations. In doing so the results of the data might not function appropriately in the real world. However, depending on the advancement of the simulation process, it might still perform exceptionally well. This report, however, is simply a proof of concept so the created final model won't need to be utilized in a production environment. So even though the dataset might not be suitable or a business, for the sake of this report it works well.

3.2 Categoric Target

When performing classification, the model looks to output a category. In most cases this will either be a 0 or a 1. Without this the model wouldn't know what to do and you would be better off utilizing clustering instead. Logistic regression, however, is a classification algorithm that does require a binary output. Since logistic regression is mandatory for this report, so is the binary target.

The dataset in question does have a target we can utilize. Each record is appended with a feature referred to as [isFraud]. This feature is a one hot encoded variable that is either 0 for legit or 1 for fraud. This means that the dataset is appropriate for logistic regression or any other classification model.

3.3 Enough Data

It's true that a machine learning model can pick up the trend of a model very quickly on as few as around 10 records. For this to be the case the trend needs to be extremely consistent. This isn't always the case in reality as data tends to not be perfect. Especially when referring to fraud. There are a lot of different values for the different features, such as amount or old balance.

The dataset in question has over 6 million records to utilize if needed. This is likely overkill for a machine learning model, but the option is there, making the dataset very valid in this department.

4 Performing Descriptive Analysis

The following section walks through the steps and findings regarding the descriptive analysis performed.

4.1 Sampling

The first step of the process is to make sure that the data has been sampled. If we attempt to graph 6 million records in a single attempt, we run the risk of running out of driver memory and crashing the IDE.

Earlier it was discovered that there are 8 thousand fraudulent records and 6 million legit records. For this reason, we only want to sample the legit records and preserve 100% of the fraudulent records. To achieve this, we separate the fraudulent records from the legit records. Following this we sample 500 000 records from the legit DataFrame.

```
fraudulentRecords = data[data['isFraud'] == 1]
legitRecords = data[data['isFraud'] == 0]
legitRecords = legitRecords.sample(n=500000,random_state=1)
```

```
There are 8213 fraudulent records.
There are 500000 legitimate records.
```

Figure 2 fraud and legit distribution

When comparing these two different types of records we cannot graph them on the same figure. The size difference wouldn't wok on a histogram or bar chart as there would be an overwhelming number of legitimate records. Instead, we will graph them individually and compare the distributions.

4.2 Visualising the Data

The following section walks through all the different graphs that were created when performing the visual analysis.

4.2.1 Transaction Types

Let's begin by comparing the different types of transactions for the legit records and fraudulent records. The focus here will be on the distribution of each.

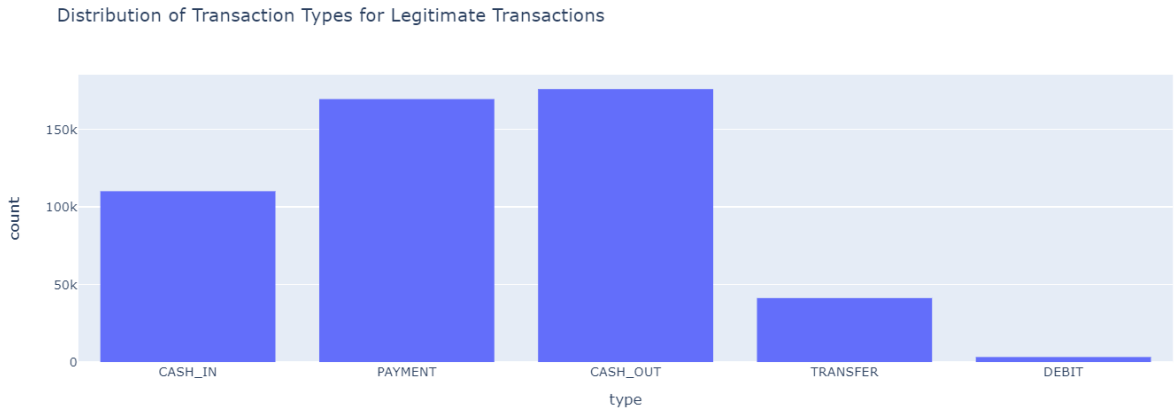

Figure 3 Distribution of Transaction types for legit records

We see that there are not a lot of debit orders and that the most common type of transfer seems to be the cash out option where people take money out of their account. We also see that transfers which involve sending money from account to account isn't vey popular in comparison and debit transactions are the quite rare. Let's now take a look at the fraudulent transactions.

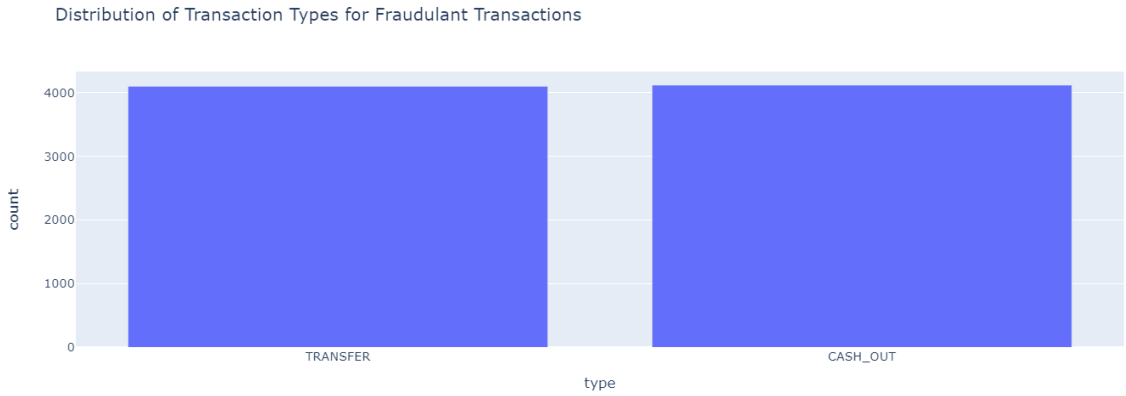

Figure 4 Distribution of Transaction types for fraudulent records

Now this is very interesting. We see that transfer and cash out are the only transaction types used in this fraudulent technique. We can conclude from this data that the process involves malicious people getting into someone's account, transferring it to another account, and then withdrawing the money in cash. The fact that these two methods have almost identical counts shows that the processes are linked. Adding a time-based element to this data could very realistically improve the accuracy of this model already, as it would show that a transfer preceded a withdrawal, but the goal of this model is to pick up on the fraud during the transfer stage. So, time will be negated here.

4.2.2 Amount Distributions

Next, we will look at what amounts are common with the different types of transactions. Again, we will be looking at the distributions of the data.

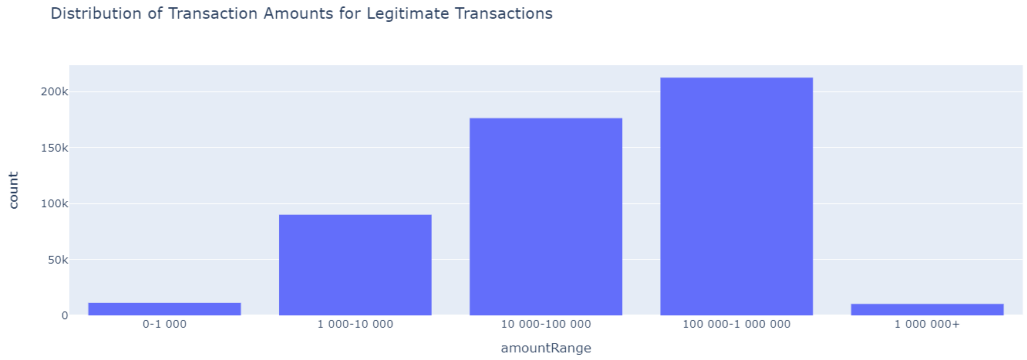

Figure 5 Distribution of transaction amounts for legitimate transactions

Amount Range	Percentage
100 000-1 000 000	0.424930
10 000-100 000	0.352506
1 000-10 000	0.179894
0-1 000	0.022342
1 000 000+	0.020328

Figure 6 Amount percentages for legitimate records

We see that the vast majority of legitimate records had amounts between 100 000 and 1 000 000. With very few records over 1 000 000 and under 1 000. This is very strange because one would assume that there would be quite a few records under 1 000. It's safe to assume then that either the simulated records are a bit weird, or the currency isn't in dollars. The currency would have to be something much weaker than the dollar, a currency where 1 000 of it isn't considered a lot. Regardless, we can proceed assuming there are no errors and will now look at the fraud records.

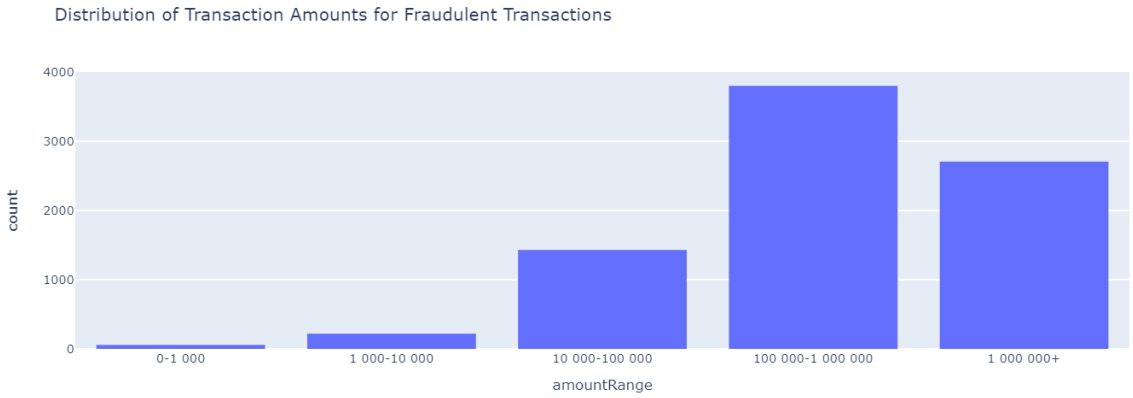

Figure 7 Distribution of transaction amounts for fraudulent transactions

Amount Range	Percentage
100 000-1 000 000	0.462681
1 000 000+	0.329478
10 000-100 000	0.173992
1 000-10 000	0.026787
0-1 000	0.007062

Figure 8 Amount percentages for fraudulent records

Now we start to see some interesting changes. We see that transactions between 0 and 1 000 became even more rare and the rare 1 000 000+ amount became the second highest amount. The 100 000 to 1 000 000 range is still dominant, but this was something to be expected.

Additionally, we see that these people who commit these fraudulent transactions don't want to waste their time with small amounts, especially if the currency is quite weak.

4.2.3 Amounts per Transaction

Next, we can take a look at the distribution of amounts relative to their respective transaction type.

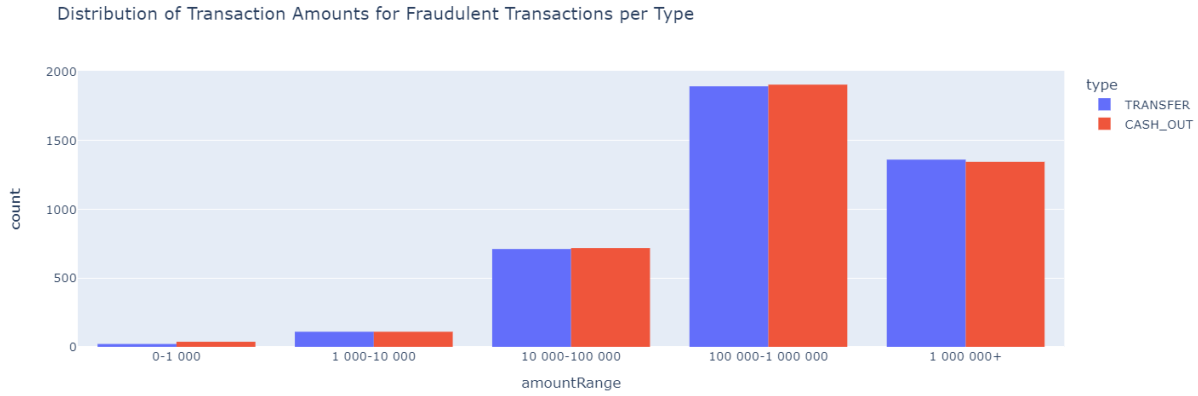

Figure 9 Distribution of Transaction amounts for fraudulent transactions per type

The biggest thing we see from this graph is that transfers and cash out are very closely linked. This further emphasises the idea that people get a hold of someone's account, transfer money to a different account the malicious person has control over, and then withdraw the money. One thing to note however, is that there is quite a prominent amount of cash out transactions being placed for the fraudulent transactions over a million. Let's take a look at the legitimate transactions to see if this differs.

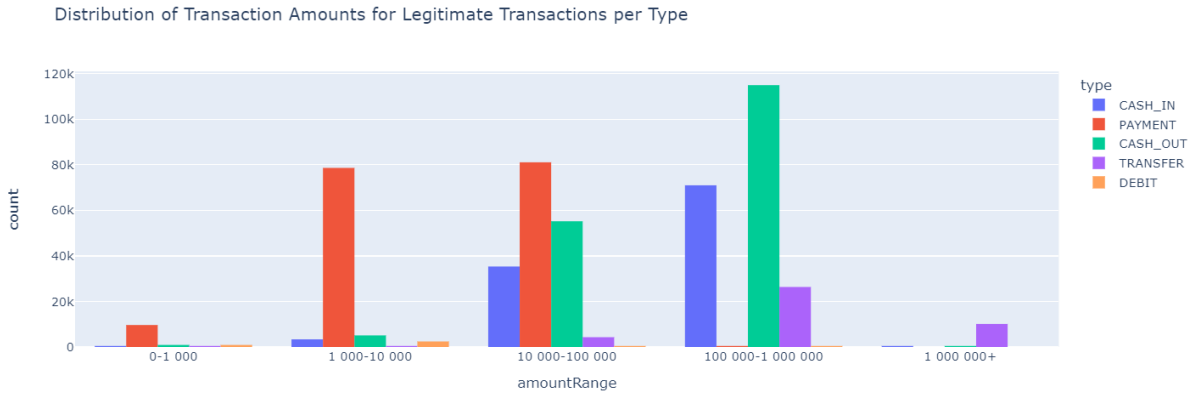

Figure 10 Distribution of Transaction amounts for legit transactions per type

There is a lot of information we can get from this graph but the thing that seems extremely important is the fact that there seems to be extremely few cash out transactions above 1 million for the legit records. This is in direct contrast to the fraudulent records which had quite a few cash out transactions.

From this we can summarize that most cash out transactions above 1 million currency is likely fraudulent, as people don't tend to want to legitimately withdraw a million of whatever currency this is.

4.2.4 Balance of Account That Received the Money

Here we will look at the difference between the accounts that received the money for both legitimate and non-legitimate transactions. Keep in mind this is the account that the hacker would deposit money into and withdraw from.

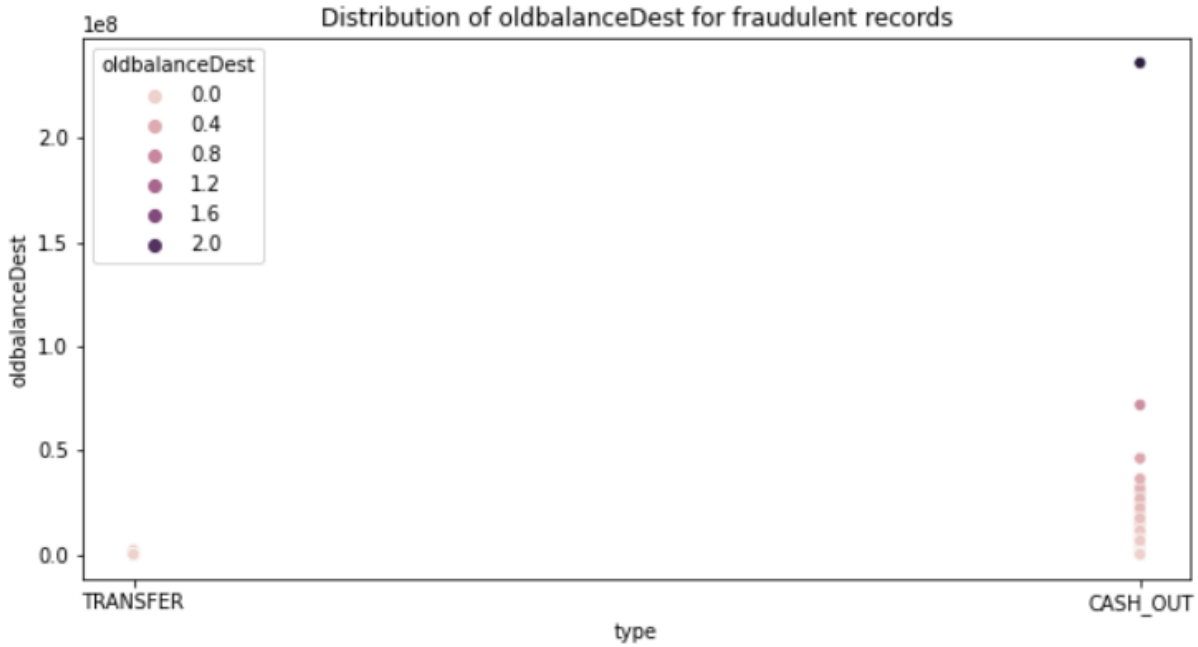

Figure 11 Distribution of destination accounts original balance from fraudulent records

We see that with transfers, the original account almost always had a balance of 0 before the transaction. This means that the account was likely created to be a burner account. There are a few cases where the account had a little bit it was probably just a test amount like 10 currencies to see if the account was working. Let's take a look at how the legit records compared.

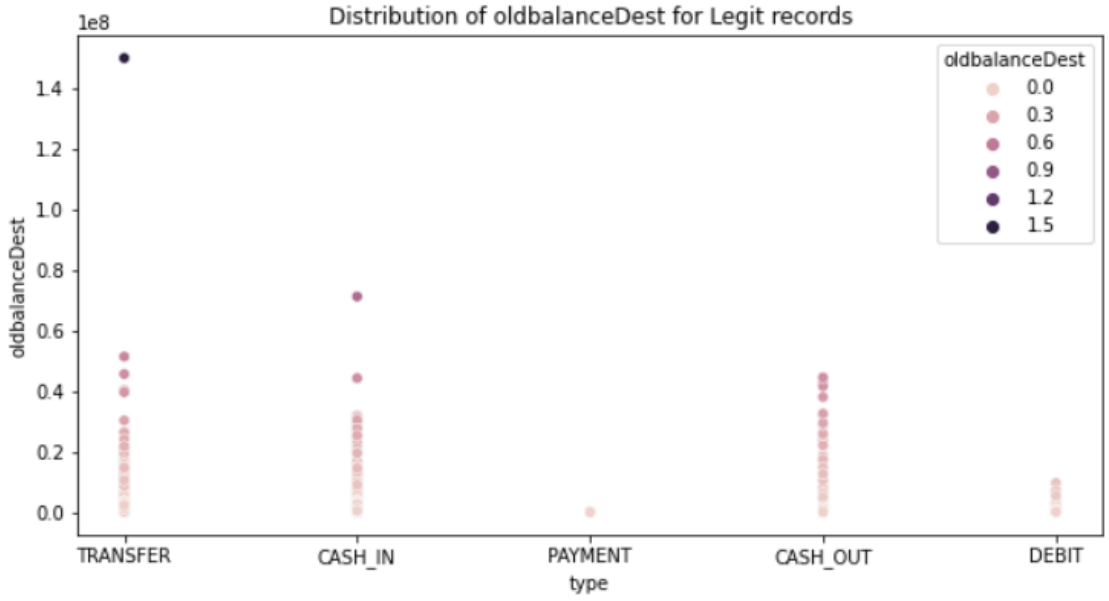

Figure 12 Distribution of destination accounts original balance from fraudulent records

Suddenly we see a very different story for the legitimate records. When someone transfers money to a legitimate account, that account very likely will have a fair amount of money in it. This is normal as people tend to have money in accounts that they are using. So, in contrast to the fraudulent accounts having very little money in them, the legitimate accounts tend to have a decent amount. Understandably so, we see that there is the exact same pattern for the new balance of the destination accounts after the transaction, this is because it is a direct before and after mirror of the transaction.

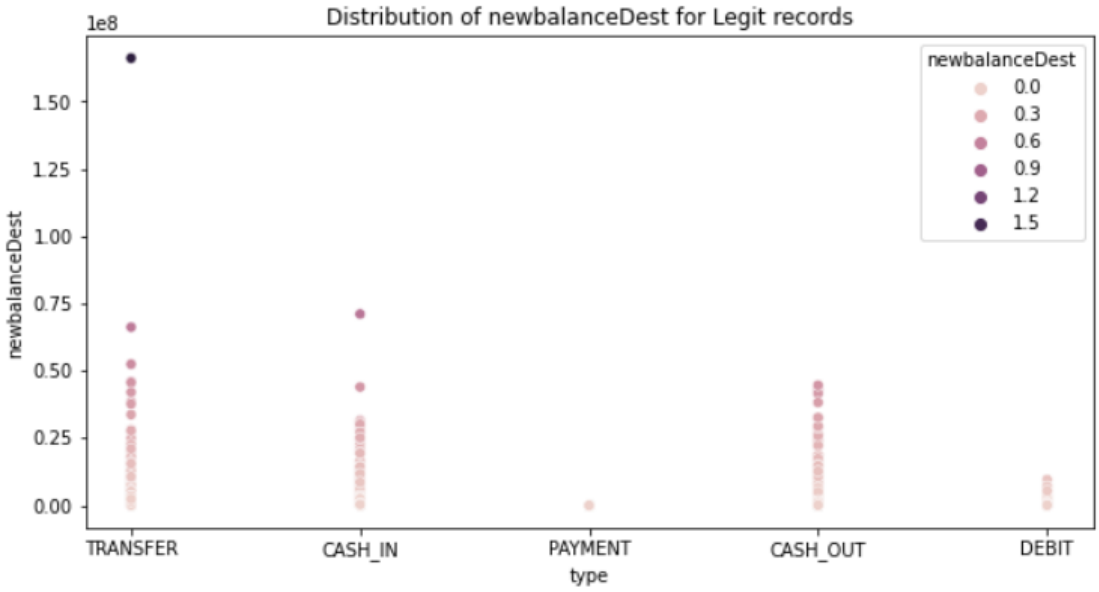

Figure 13 Distribution of the destination accounts new balance after the transaction for legit records

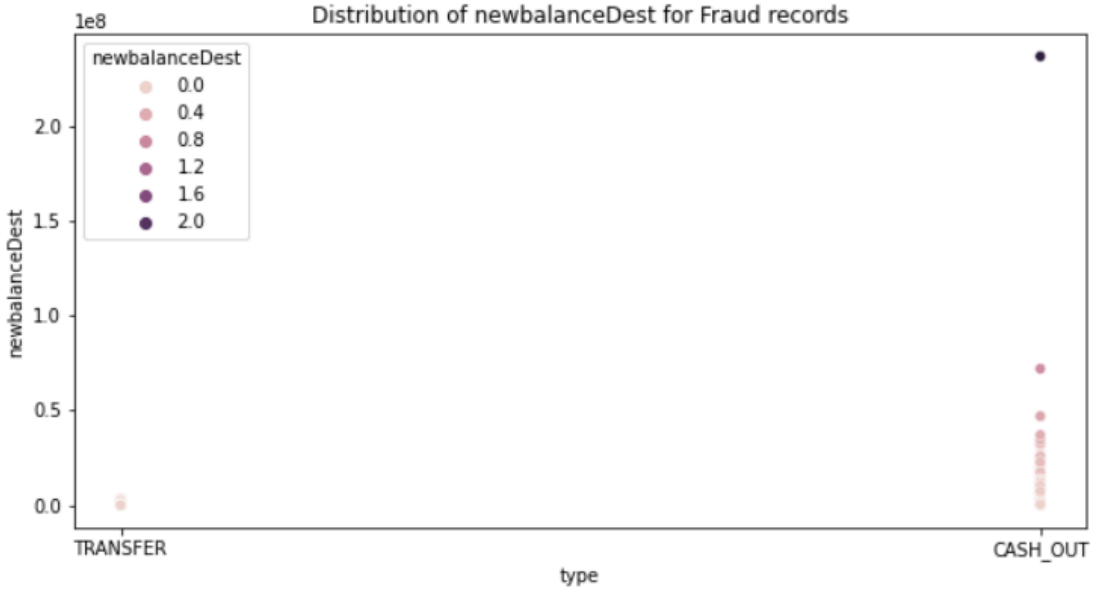

Figure 14 Distribution of the destination accounts new balance after the transaction for legit records

4.2.5 Balance of Account That Sent the Money

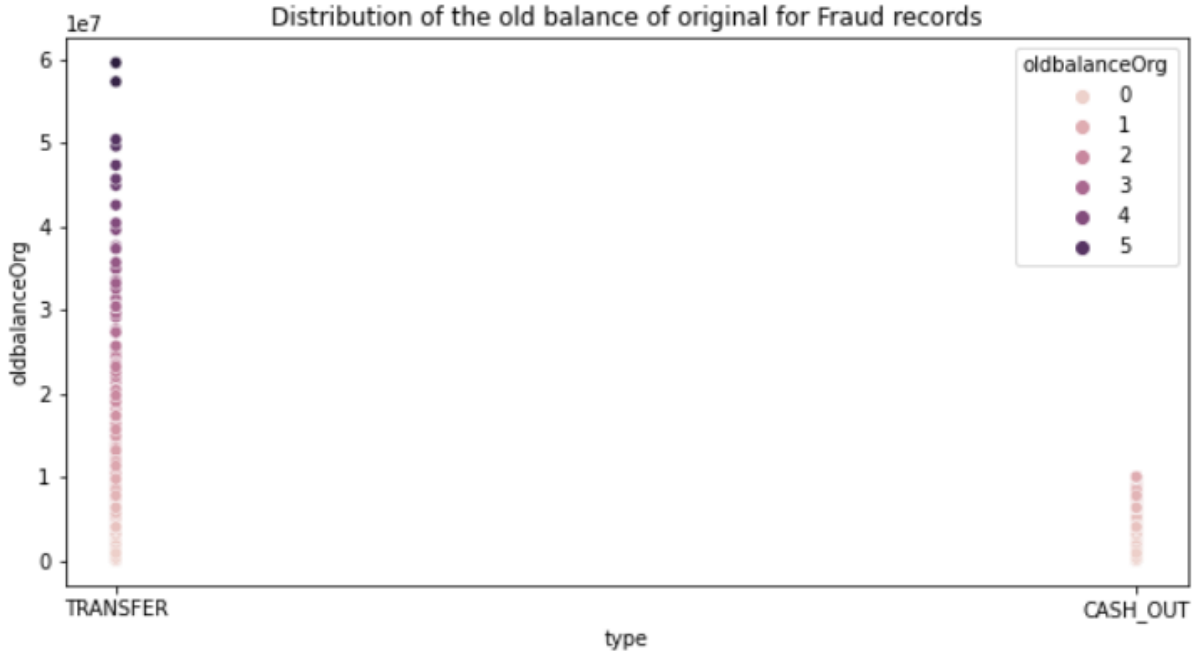

Figure 15 Distribution of the old balance for the senders account for fraud records

We notice that the amounts in the accounts of the sending account is often quite high. This is to be expected as these are often real accounts attached to real people to use them for day-to-day purchases. Let's see if there is a difference for legit records.

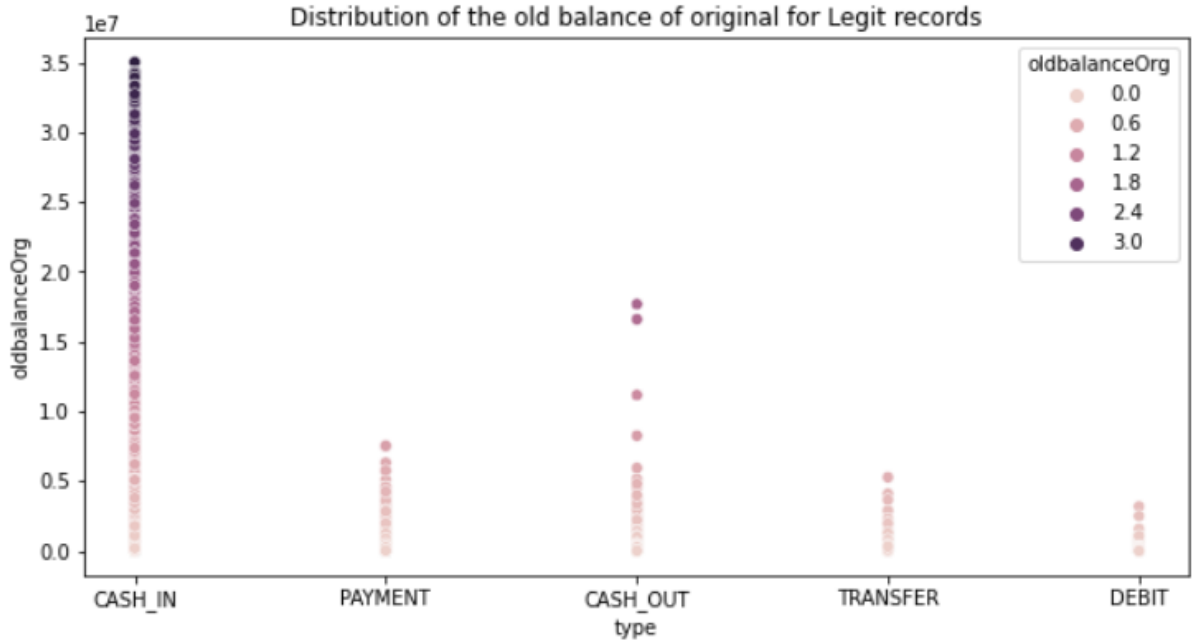

Figure 16 Distribution of the old balance for the senders account for fraud records

Here we see something very interesting, we see that the original balance for transfers is often quite low, which is in contrast to the fraud records. This shows that people who have large amounts of money tend to not perform transfers for some or other reason. Instead, they tend

to opt for the cash in transaction type. Like the previous model, we are likely to see a mirror of these graphs with the new balance.

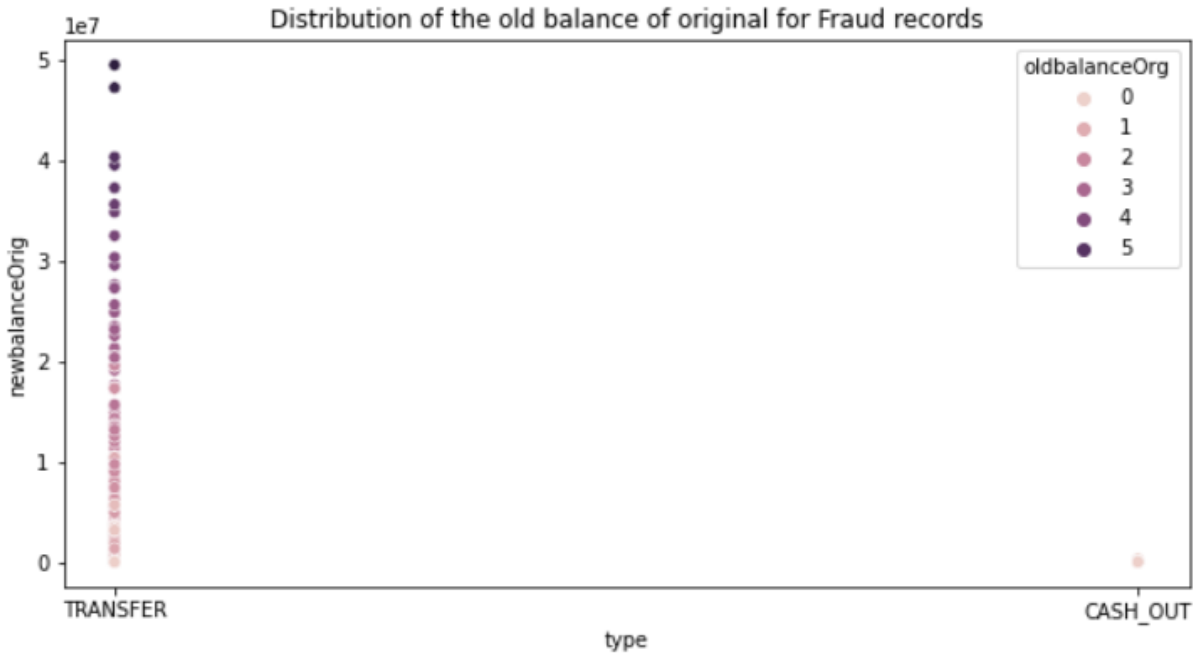

Figure 17 Distribution of the sender's accounts new balance after the transaction for legit records

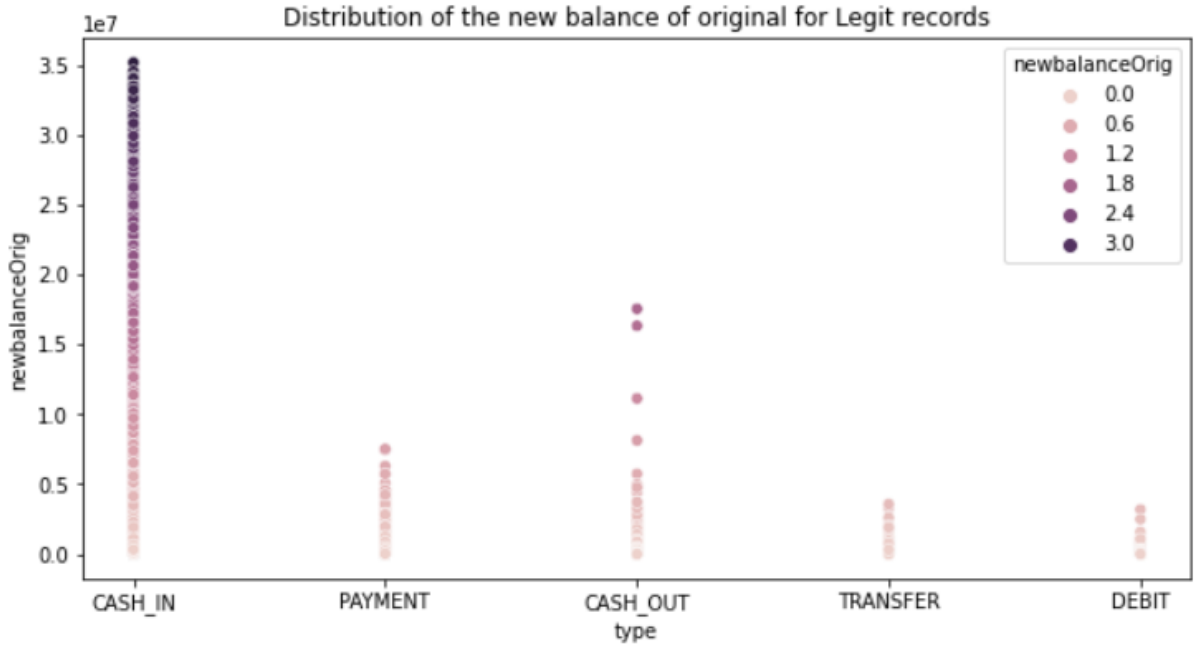

Figure 18 Distribution of the sender's accounts new balance after the transaction for fraud records

4.2.6 Final Digit Distribution

Now let's take a look at the final digits for both legit and fraudulent records.

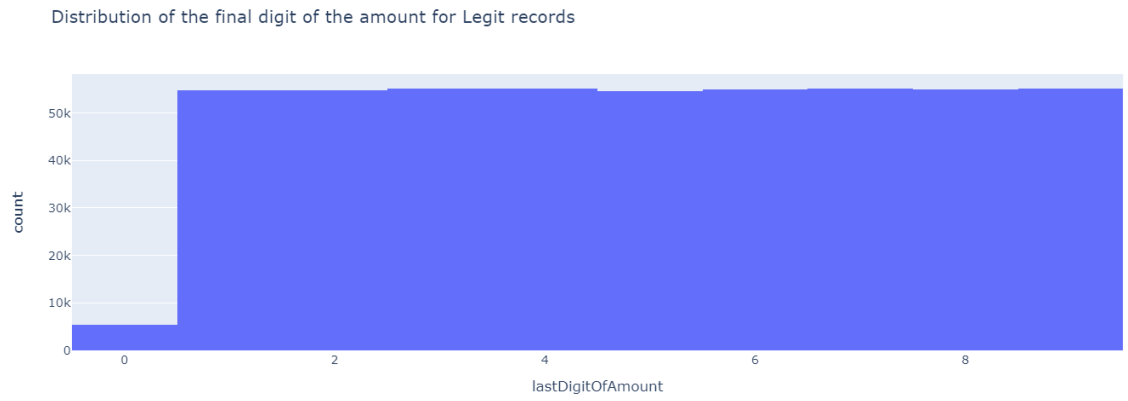

Figure 19 Distribution of the final digit for legit records

We see that with legit records there are very few records that end in a 0. This is because real prices very often aren't divisible by 10. Instead of something costing 10 dollars, instead it will cost 9,99 dollars to make the price more appealing. Let's see if there is a difference with the fraudulent records.

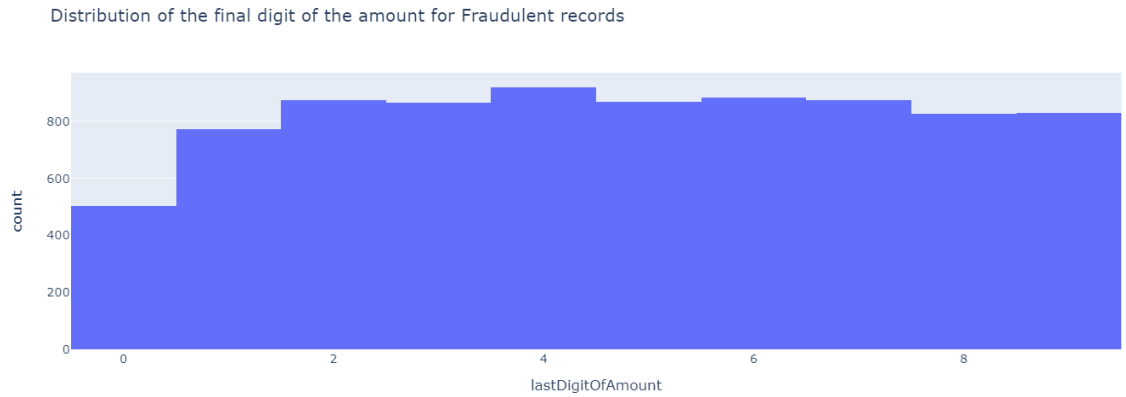

Figure 20 Distribution of the final digit for fraud records

Sure enough, with fraud transactions, there are a lot more transactions that end in a number divisible by 10. This is because you aren't actually buying something that needs to have an appealing price. The hacker is more likely to simply transfer a 100 000 straight from the account. It seems, however, that some of them realise that whole number transactions look a little suspicious, so they try to be more cautious with it.

We can also compare with the different types of transactions.

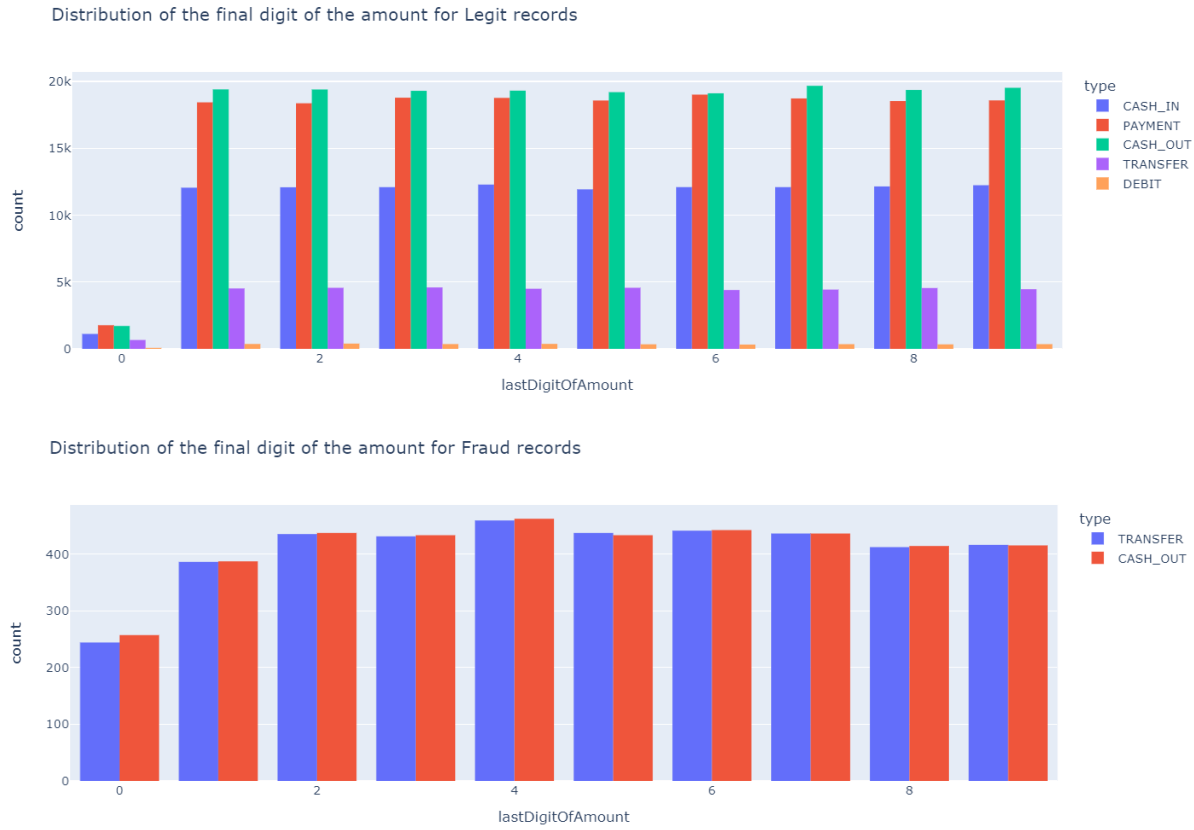

Figure 21 Distribution of the final digit of the amount in a legit transaction

We don't learn anything new from this that we didn't already know. Simply that the distribution between the different transfer types is quite balanced.

4.2.7 Correlations

Finally, let's look at the correlations between each variable and see if there is something that sticks out.

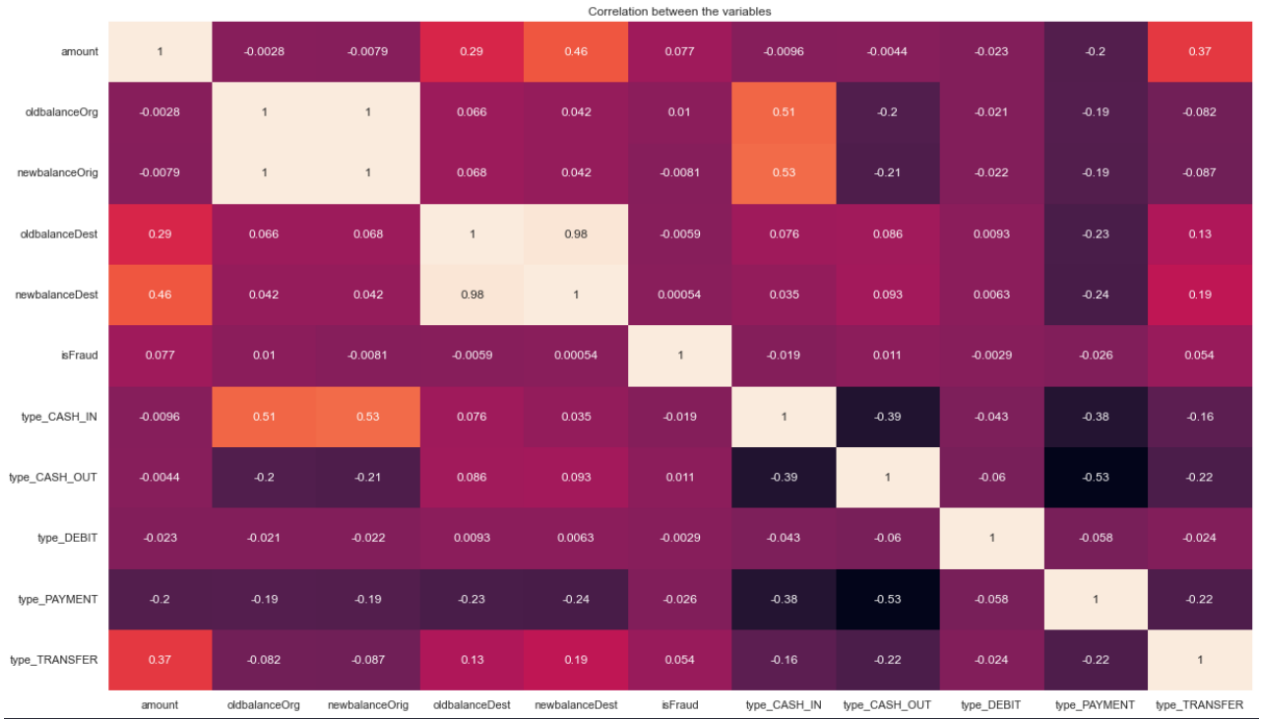

Figure 23 Correlations

There aren't any really good correlations with the target variable of isFraud. But there are some good correlations between certain variables, such as old balance and cash in. These correlations wouldn't be enough to come to a conclusion with either one variable by itself, but they do add to the models ability to perform conditional probability.

4.2.8 Summary of findings

This dataset had a lot of useful information and a lot of small details that the model is likely going to pickup on. From this dataset we learned that;

- Fraudulent transactions are almost exclusively (for this dataset) conducted by a transfer from an innocent person's account to a malicious person's account before the malicious person withdraws the money. The malicious person likely withdraws the money to prevent the transaction from being refunded. Once the cash is in their hands, the bank can't take it back.
- Fraudulent transactions tend to be a lot larger than legitimate transactions.
- Cash outs of over 1 million are almost exclusively fraudulent, likely because people don't want to have that amount in cash.
- If an account has no money in it and receives a transfer, then it is likely fraud.
- If an account has a lot of money in it and performs a transaction, it is likely fraud.
- If an amount transferred end in a 0, it is likely fraud.

None of these findings alone is enough to accuse a transaction of fraud, but if a new record satisfies quite a few of these requirements, then it is likely fraud. This is where the machine learning model can perform conditional probability to make a very accurate model.

5 Data Cleaning

The following section will describe the process taken to cleaning and processing the data to make it viable for a model to work with.

5.1 MinMaxScaler

The MinMaxScaler will take numeric data and scales it down between 0 and 1 to make it easier for the computer to read and interpret. The important thing to note is that the distribution of the data will not be changed, the values will simply be made smaller.

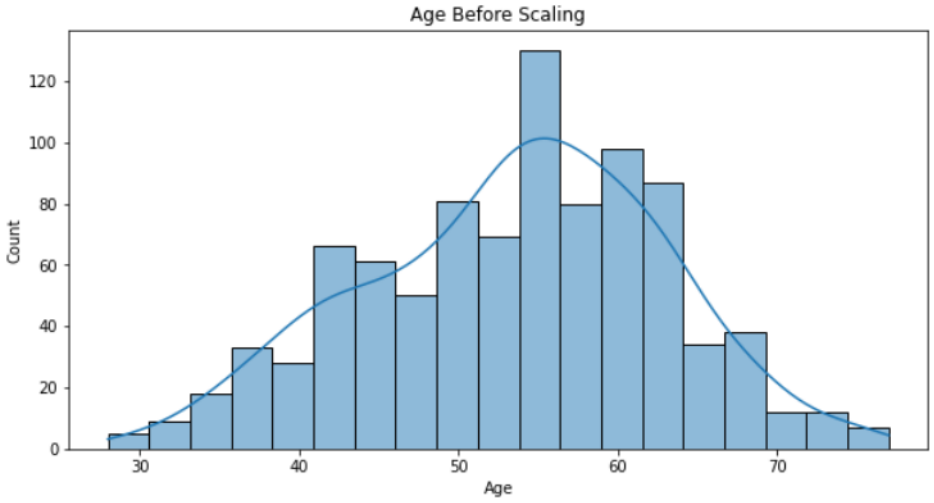

Figure 25 Data before it is scaled

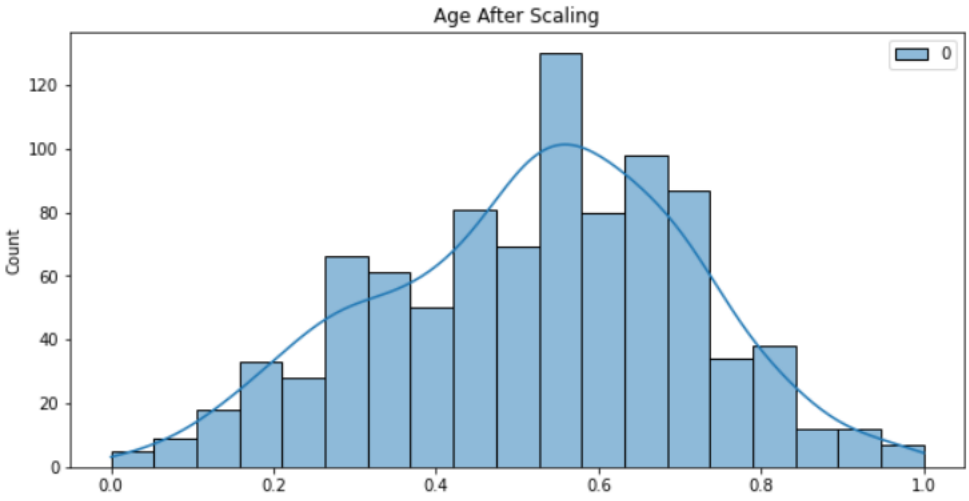

Figure 24 Data after being scaled

We notice that the values for age have now been scaled to values between 0 and 1.

5.2 Sample the Data

We start off by sampling the data to test our models. Working with too much data would make the training time for the models unrealistic for this report. Additionally, machine learning models typically don't require too much data anyway. If there seems to be underfitting then we can always increase the number of records.

```
------------------------------------------------------------------------------
Before Sampling:
------------------------------------------------------------------------------
There are 8213 fraudulent records.
There are 6354407 legitimate records.



------------------------------------------------------------------------------
After Sampling:
------------------------------------------------------------------------------
There are 8213 fraudulent records.
There are 20000 legitimate records.
```

Figure 26 Before and after sampling

5.3 Drop Null Values

After sampling, we can make sure that there are no null values within the dataset. If there are null values, then we need to drop them as they will break our model which relies on each feature having a value. Luckily this dataset did not host any null values.

5.4 Create Dummy Values

Any categoric features that we plan to use will instead have each unique converted to its own one hot encoded column.

type
PAYMENT
PAYMENT
TRANSFER
CASH_OUT
PAYMENT
PAYMENT
PAYMENT
PAYMENT
PAYMENT
DEBIT

Figure 28 Before Dummies

type_CASH_IN	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER
0	0	0	0	1
0	1	0	0	0
0	0	0	0	1
0	1	0	0	0
0	0	0	0	1
...	...	...	...	...
0	0	0	1	0
0	0	0	0	1
0	1	0	0	0
0	0	0	1	0
0	0	0	0	1

Figure 27 After dummies

5.5 SMOTE

We see that there are a lot more legit records than fraudulent records which is a problem. This means that the model runs the risk of not being able to grasp the concept of what is fraudulent as there aren't enough records for it. If 99% of the records are 0, and the model only ever guesses 0, then the model accuracy will be 99% but the model is completely useless. So, to combat this we introduce SMOTE, which will generate fake fraudulent records based on averages between the real records.

5.6 Train and Test Sets

The next step of the process is to break the data into training data and testing data. The training data will be used to create and validate the model, well the testing data will be used to give an unbiased assessment of how well the model performs. It's important to not let the model see the testing data until you plan to assess it. The testing data needs to act as a simulation of real world, never before seen data.

6 Model Creation

The following section will summarize the process taken to creating the best possible model to perform fraud detection on the current dataset.

6.1 Logistic Regression

Logistic regression is a two-level outcome which takes the natural logarithm of odds as predictors (LaValley, 2008). The logistic regression model follows the concept of the sigmoid. To demonstrate this, we start with points that follow a straight path.

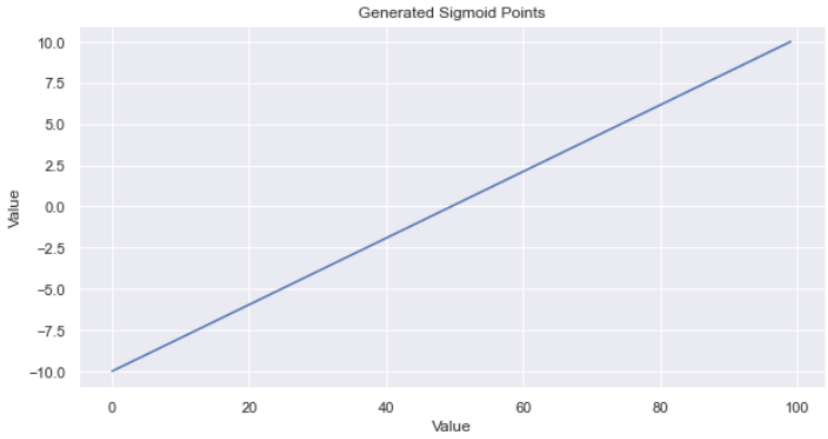

Figure 29 Straight Line points before sigmoid function

Next, we apply the formula of $s(x) = \frac{1}{1+e^{-x}}$ to each data point and then graph it. The result is as follows.

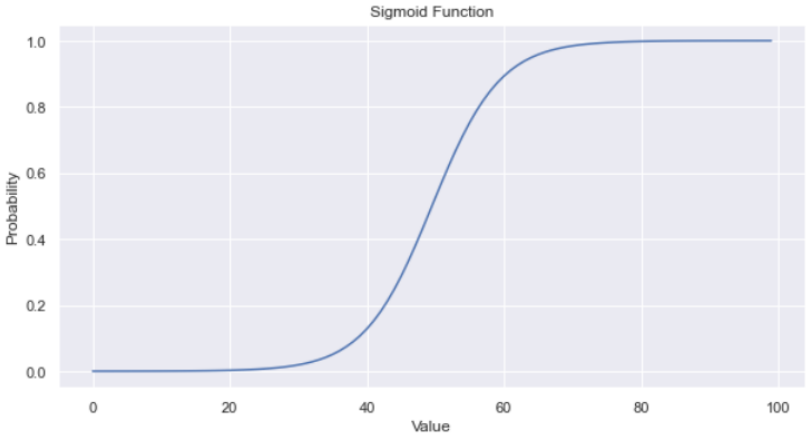

Figure 30 Sigmoid Graph after sigmoid function

As we see above, the sigmoid points approach 1 and 0 in either direction but will never actually reach it. So instead, we take the probability values from this graph. For instance, if the value is 80, there is a high probability that the prediction is a 1. If the value is 40 then there is a high probability that the prediction is a 0. This forms the basis of logistic regression.

To perform logistic regression from scratch there are a few steps to follow. We first next to give a function the X and Y values. We then take the number of samples and the number of features.

```
n_samples, n_features = X.shape
```

We then need to create some weights, which start at 0 before be apply gradient decent. We need an array of 0s for each feature. We also need to set the initial bias to 0.

```
weights = np.zeros(n_features)
bias = 0
```

The following code gets iterated as many times as the person deems necessary when fitting the data. We need to get a linear prediction which works on the following formula $LP = (x \cdot w) + B$. Following this step, we need to throw the linear predictions into a sigmoid function to get a sigmoid back as the initial predictions.

```
linear_pred = np.dot(X, weights) + bias
predictions = sigmoid(linear_pred)
```

The next step is to get the derivatives of the weights with the formula:

$$derivOfWeights = \left(\frac{1}{Number\ of\ samples}\right) * (transposed(X) * (Predictions - y))$$

Following that we get the derivative of the bias:

$$derivOfBias = \left(\frac{1}{Number\ of\ samples}\right) * (sum((Predictions - y))$$

```
dw = (1/n_samples) * np.dot(X.T, (predictions - y))
db = (1/n_samples) * np.sum(predictions-y)
```

Next, we update the weight and bias by subtracting the product of the derivatives and the learning rate.

```
weights = weights - lr*dw
bias = bias - lr*db
```

The whole class will look as follows.

```
def sigmoid(x): #Sigmoid function
    return 1/(1+np.exp(-x)) #Sigmoid function

class LogisticRegressionScratch(): #Logistic Regression Class

    def __init__(self, lr=0.001, n_iters=1000): #Initialize the class
        self.lr = lr #Learning rate
        self.n_iters = n_iters #Number of iterations
        self.weights = None #Weights
        self.bias = None #Bias

    def fit(self, X, y): #Fit the data
        n_samples, n_features = X.shape #Get the number of samples and
features
        self.weights = np.zeros(n_features) #Initialize the weights
        self.bias = 0 #Initialize the bias

        for _ in range(self.n_iters): #For each iteration
            linear_pred = np.dot(X, self.weights) + self.bias #Get the linear
prediction
            predictions = sigmoid(linear_pred) #Get the predictions

            dw = (1/n_samples) * np.dot(X.T, (predictions - y)) #Get the
derivative of the weights
            db = (1/n_samples) * np.sum(predictions-y) #Get the derivative of
the bias

            self.weights = self.weights - self.lr*dw #Update the weights
            self.bias = self.bias - self.lr*db #Update the bias

    def predict(self, X): #Predict the data
        linear_pred = np.dot(X, self.weights) + self.bias #Get the linear
prediction
        y_pred = sigmoid(linear_pred) #Get the predictions
        class_pred = [0 if y <= 0.5 else 1 for y in y_pred] #Get the class
predictions
        return class_pred #Return the class predictions
```

Source: (AssemblyAi, 2022)

Now we can introduce our training data and then test to see how well the model performed.

	precision	recall	f1-score	support
0	0.93	0.62	0.74	3935
1	0.72	0.96	0.82	4065
accuracy			0.79	8000

Figure 31 Classification score for logistic regression

We see that the overall accuracy of the model isn't too bad, but the recall of the legit records is a bit disappointing. This is where we can look at hyperparameters and testing to see if there are any other models which could potentially gain a better grasp of the data. But first we need to establish a decent technique for verifying the data.

6.2 Model Verification

Once we have a prediction there is always the risk of the accuracy score being a complete fluke. To combat this, we will use the k-folds validation technique which follows the logic of "Let's test it a bunch of times and see if the score is consistent".

The method breaks the dataset into k number of folds and then uses 1-fold to test the model and the remaining folds to create it (Koehrsen, 2018). It grabs the accuracy and then moves onto using the next fold to test the model. This repeats until all the folds have had a chance to act as the testing data.

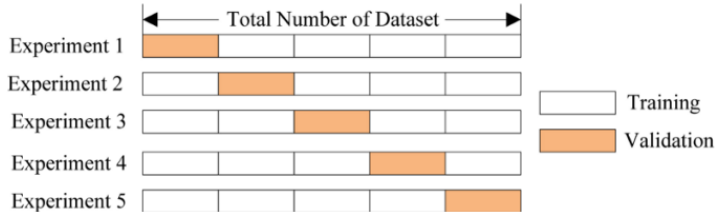

Figure 32 K-folds by (Koehrsen, 2018)

6.3 Hyper Parameters

Next, we are going to try and apply hyper-parameters to our logistic regression to see if there are any parameters which results in a higher accuracy. We are going to test to see which value of C is the best for logistic regression and then train the data on that. For this we will utilize grid search which loops through all the parameters and selects the best one based on accuracy.

```
C = [0.001, 0.01, 0.1, 1, 100] #C values
LogisticRegressionParamGrid = [{'C': C}] #Create the grid
LGR_GRID = GridSearchCV(LogisticRegression(max_iter=4000),
LogisticRegressionParamGrid, cv=5, scoring='accuracy')
LGR_GRID.fit(X_train, y_train)
```

The output is that the best accuracy was achieved when C was set to 0.001.

After training the model on these parameters we see that there really wasn't much difference between the accuracy where C is set to 1 (default) and when it was set to 0.001.

	precision	recall	f1-score	support
0	1.00	0.57	0.72	3935
1	0.70	1.00	0.83	4065

Figure 33 Logistic regression with hyper parameters

6.4 Selecting the Best Model

We could manually check to see how each model performed, and in a production environment that is probably the recommended way of doing it since you can then play around with parameters and small tweaks without sitting through the evaluation of each model. But for the this reports we will simply run a method that will cross-validate each model again our data and then return it.

```
Models = {'RF': RandomForestClassifier(),'KNN': KNeighborsClassifier(),'NB':
GaussianNB(),'SVC': SVC(),'LGR': LogisticRegression(max_iter=4000)}

def getBestModel(Models, X_train, y_train, X_test, y_test):
    Best_Model = 'LGR'
    Highest_Score = 0
    report = {}
    for key in Models:
        pipeline = make_pipeline(scaler, Models[key])
        scores = cross_val_score(pipeline, X_train, y_train, cv=5,
scoring='accuracy').mean()
        report[key] = scores
        if scores > Highest_Score:
            Best_Model = key
            Highest_Score = scores
    return Best_Model, Highest_Score, report

Best_Model, Highest_Score, report = getBestModel(Models, X_train, y_train,
X_test, y_test)
```

```
The Best model was RF with a score of 0.9950625000000001
The scores were
{'RF': 0.9950625000000001, 'KNN': 0.9808125000000001, 'NB': 0.781, 'SVC': 0.83396875, 'LGR': 0.8044687500000001}
```

Figure 34 Model Scores

We see that the Random Forest Classifier is blowing the other models out of the water. The K-nearest neighbour is also performing quite well, which isn't surprising as it's a good model to cluster people together. From now on we will be utilizing the Random Forest model.

6.5 Why the Random Forest Model outperformed Logistic Regression so extremely

The random forest classification algorithm gathers multiple decision trees which are each trained on slightly different segments of the data. These decision trees are all given the same input data and then asked to guess what the classification should be. The majority vote is taken (Pal, 2005).

6.5.1 How Random Forests Work

The trees work on a hierarchical concept where the data travels down the branches of the trees until they reach a leaf node which holds a classification output.

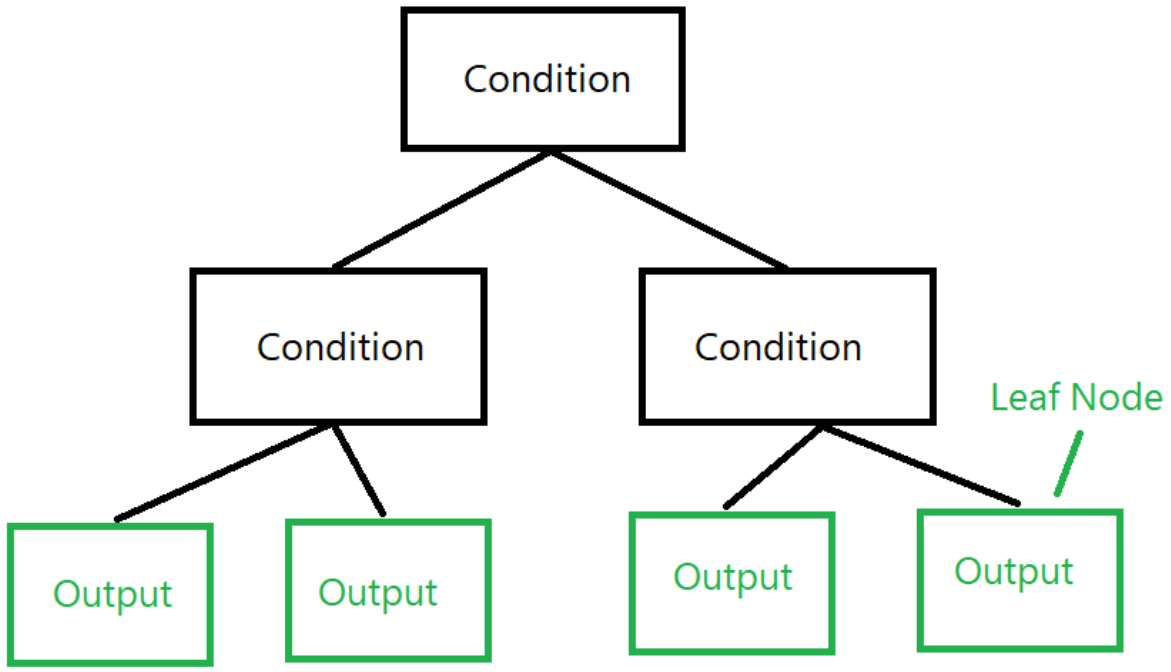

Figure 35 Decision Tree

6.5.2 The Obvious Flaw with Logistic Regression

Logistic regression classifies data by whether or not it is bigger or smaller than a certain threshold. So, assume the diagram below.

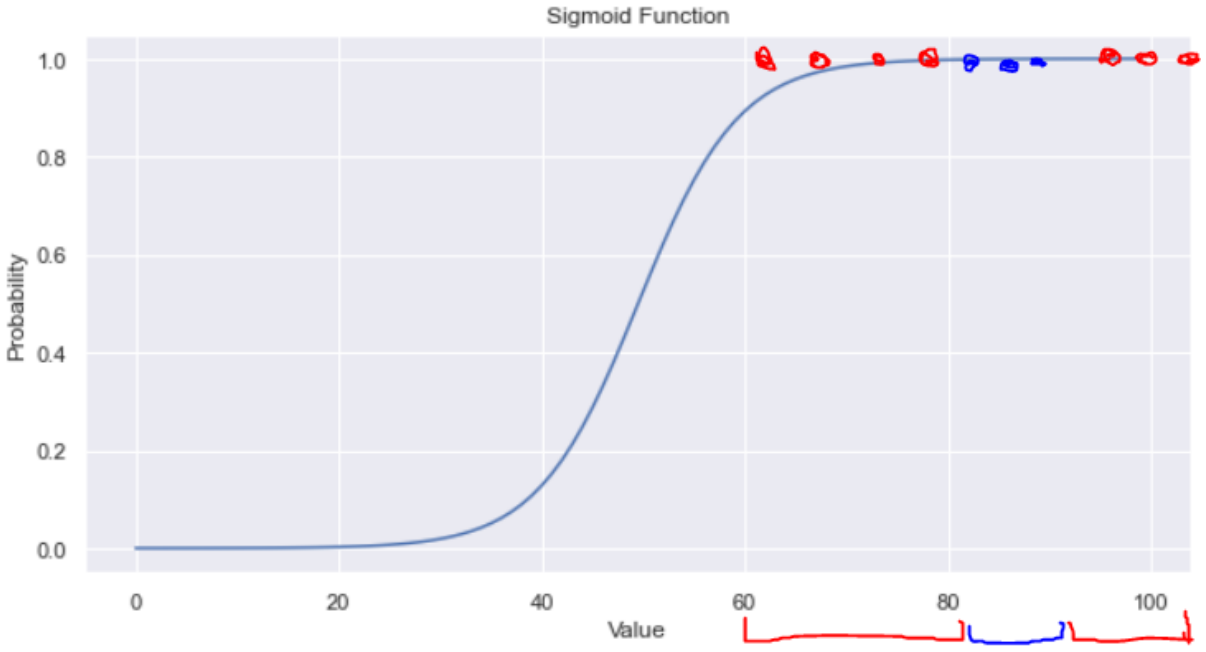

Figure 36 Logistics flaw

We assume that between 60 and 80 the classification is red and between 90 and 100 the classification is also red. But right in the middle, if it's between 80 and 90, the classification needs to be blue, the logistic regression model can't pick that up. The logistic regression model

simply knows that above 50 the value is 1. It is unable to pick up on those small details that might deviate from that concept.

6.5.3 Why the Random Forest isn't Affected by This

The random forest model works like a nested if statement. Which means that if you allow it infinite leaf nodes it could theoretically have an accurate classification for every possible value. But most importantly nested if statements allow for conditions to be dependent on previous conditions.

The above example could be solved as following:

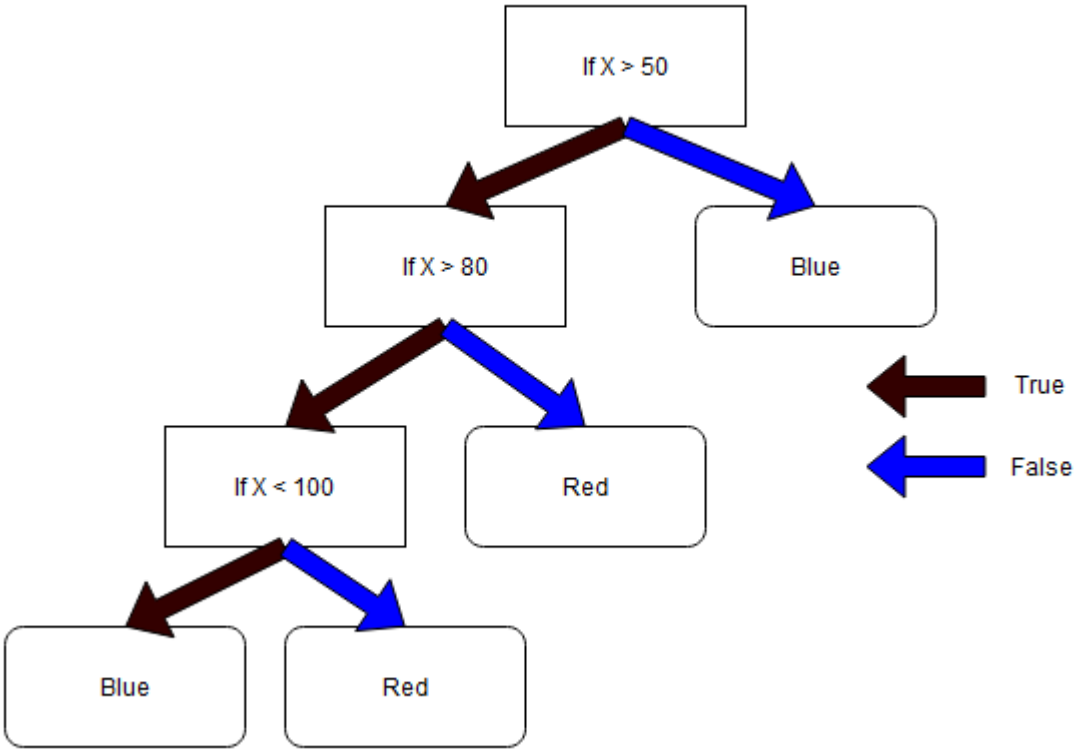

Figure 37 How Random Forest could solve the problem with logistic regression

6.6 Overfitting and Underfitting Considerations

Overfitting is when the model is too closely fit to the training data which prevents the model from accurately predicting unseen data as it is too reliant on the data before it. An underfit model, however, is a model that is unable to grasp the trend of the data (Muller & Guido, 2017).

6.6.1 Overfitting

The best way to deal with overfitting it to make sure that the data you are using to train the model is representative of production data. If the data is constructed manually, too much bias might be added into it and the model may suffer as the result. This is especially difficult to detect as the testing data will likely also have these bias in them.

This is actually a big problem for the model we have created. It isn't guaranteed, but since the data is simulated, it might be the case. If this was a real model going into production, then it would have to be tested against real data.

One way to avoid overfitting, especially with the Random Forest Classification is to reduce the number of nodes the trees can grow. This is known as pruning the tree, the correct number of nodes was discovered during hyperparameters and turns out to be no limit. Which means that either the data is already overfit, or the records are simply very consistent.

Another consideration is that SMOTE might make overfitting worse as it will duplicate the bias between the records. Further strengthening the unrealistic records.

The only real way to test for overfitting is to make sure that you are getting reliable testing data.

6.6.2 Underfitting

The biggest culprit of underfitting is having bad data to begin with. If there is no link between the input variables and whether it is fraud or not, then the model isn't going to be able to predict anything accurately. Luckily this doesn't seem to be a problem with this dataset as the model seems to grasp the trends and patterns quite nicely.

If the data isn't plentiful enough it can also lead to underfitting. If this is the case, then we need to get more data to work with. Machine learning models don't need too much data to begin with so in business setting this shouldn't be too difficult. Luckily for this model however, it has access to as many as 6 million records if it needs it. There is a small chance that the overwhelming majority of legit records overshadows the fraudulent records, but that can be mended with SMOTE.

If data is unclean, it can also lead to the model being underfit. Null values need to be dropped and numeric values should be scaled. It's important to make sure that the data is consistent.

Bad parameters can also lead to a model being underfit. An example would be pruning a decision tree too much to the point where is doesn't have enough nodes to accurately come to an accurate conclusion.

6.6.3 Testing for Overfitting and Underfitting

Testing for overfitting and underfitting can be done by breaking the data into training data and testing data. The training data is used to train the model while the testing data is used to simulate how the model would perform in production.

6.7 Hyperparameters for Random Forest Classification

Now that we have decided that the random forest classification algorithm is what we are going for, we can grid search it to see which parameters are the best.

```
n_estimators: 41
min_samples_split: 5
min_samples_leaf: 1
max_features: sqrt
max_depth: None
bootstrap: False
```

Figure 38 Hyperparameters for random forest

We see that the random forest model performed better when there were exactly 41 trees. The odd number is preferred as it will eliminate the chance that there will be an even number of trees saying legit and fraud. We will also introduce a much larger sample of data. **Where we train the model with 500 000 records.** We use the best parameters and then see how well the model performed.

```
The average score was 0.9979125
The scores were [0.997825  0.9979375 0.997875  0.9980375 0.9978875]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     49748
           1       1.00      1.00      1.00     50252

    accuracy                           1.00    100000
   macro avg       1.00      1.00      1.00    100000
weighted avg       1.00      1.00      1.00    100000
```

Figure 40 Classification score for random forest

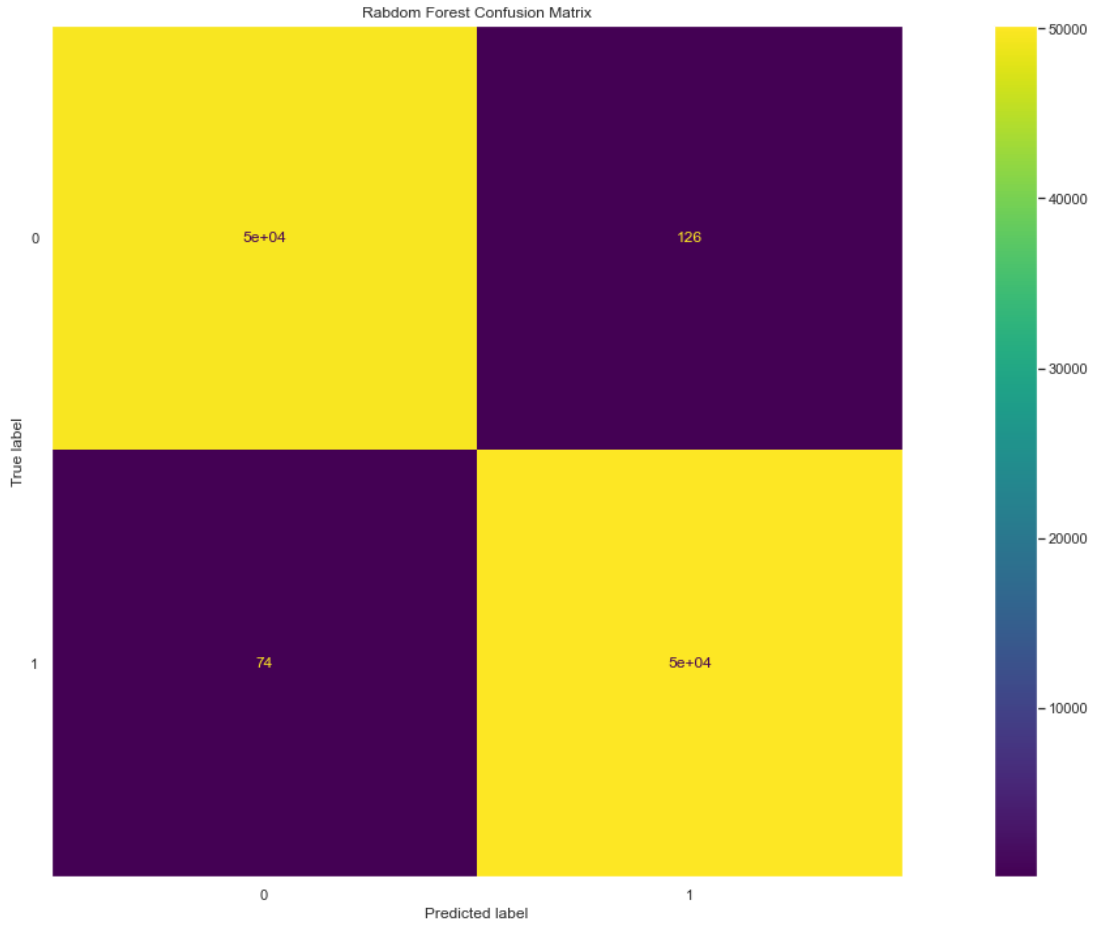

Figure 39 Confusion matrix for random forest classification

We see that the random forest model is performing extremely well. The hyperparameters increase the accuracy by about 0.002%, which in terms of a million records, is 2000 records which are now correctly identified.

Another interesting thing we can do with the random forest model is get a tally of how important each feature is.

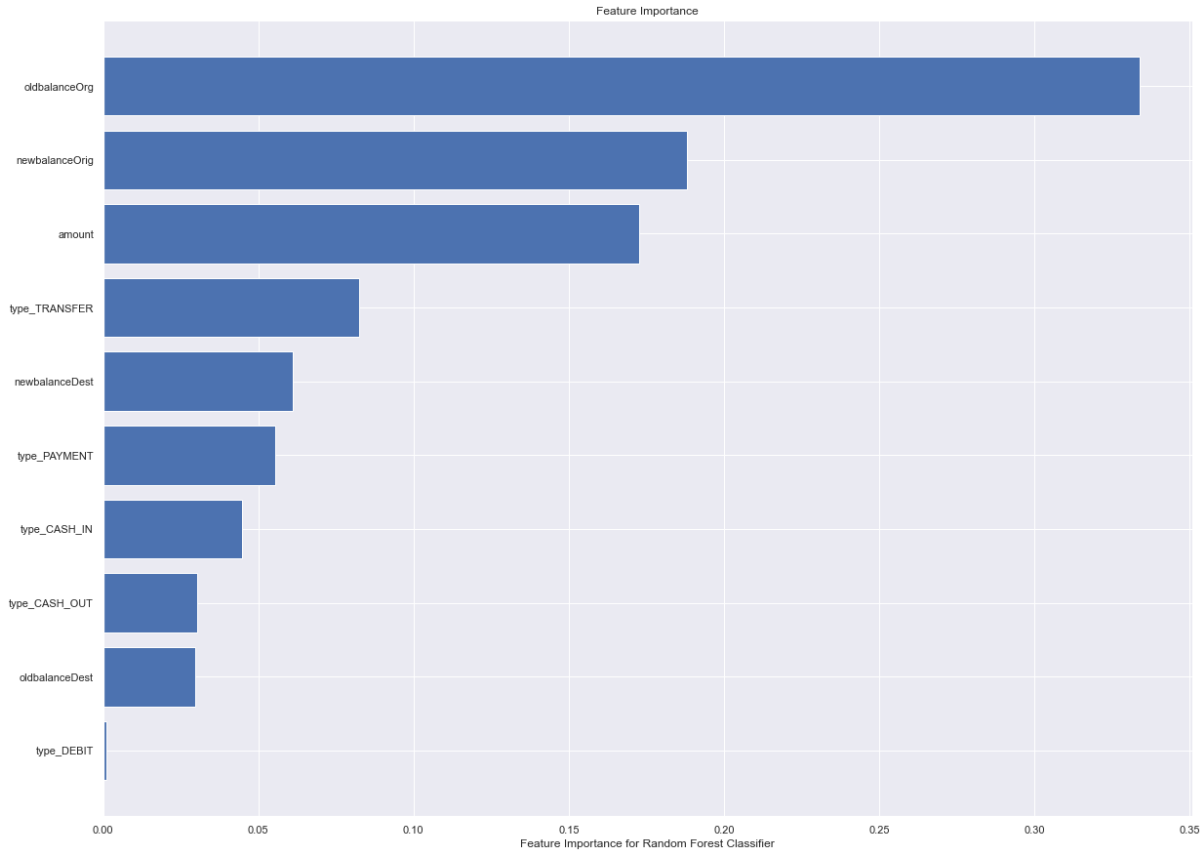

Figure 41 Feature important

We see that the old balance of the original account was the most important defining feature of if it was fraud or not. This isn't a surprise as the descriptive analysis showed that the accounts with large amount of money didn't want to perform transfers. The amount of the transaction also seemed to be very important as the fraudulent transactions tended to be quite large in comparison to the legit transactions.

7 Recommendations

The data shows that transfers where the most popular way to commit fraud. In response to this, all transfer transactions should be looked at carefully by the model. If the model suspects that it is fraud, then the transaction should be paused, and the owner should be contacted about it. If the cancellation is too late, then the person who had their money stolen could be put under a lot of unnecessary stress which makes the bank look bad. Alternatively, if the fraud is stopped in time and the original account owner is contacted, this makes the bank look good. This provides the bank with a competitive advantage over the other banks which aren't utilizing this technique.

The model performs extremely well but the data can't be trusted. So, my recommendation would be to get data from a more reliable source. If the model still performs exceptionally then

let it perform prescriptive analysis and block transactions automatically. Don't wait for the transaction to happen since if the fraudulent person manages to withdraw the money, then that money is gone. The bank can't get that back with a quick undo button. So, the model has to block the transaction at the transaction stage. If it really needs to it can also block it at the cash out stage, but that could result in a lot of stress for legit cash out transactions.

Transfers often don't need to be completed immediately, but withdrawals at an ATM do. Which means if you block someone at the transfer stage it won't be a massive deal since you can just call them quickly. But if an innocent transaction gets blocked at the cash out stage, it could be very stressful for the person with people behind them in the line.

So, I recommend blocking the transfer stage as it has lower risk of causing stress.

8 Conclusion

This report walked through the creation of a model which could take a transaction and accurately detect if it was fraud or not. We discovered that the method in question was when someone got hold of another person's account and then proceeded to transfer that into their own account before withdrawing the money before the bank can undo the transaction. During the exploratory analysis we discovered that fraudulent transactions are typically transfers and they tend to be a lot larger. Cash outs over 1 million are almost exclusively fraud. If an account has no money in it and receives a transfer it is likely fraud. If an account has a lot of money in it and performs a transaction it is likely fraud. And lastly if the amount transferred ends in a 0 it is likely fraud. The logistic regression worked well but it was ultimately the random forest classification model which stole the show with its ability to be very precise. After hyperparameters the model performed almost perfectly, and if the performance persists from real data, then the model could be utilized to block fraudulent transactions before they even happen.

9 References

Akoglu, H., 2018. User's guide to correlation coefficients. *Turkish journal of emergency medicine,* 18(3), pp. 91-93.

AssemblyAi, 2022. *How to implement Logistic Regression from scratch with Python.* [Online] Available at: https://www.youtube.com/watch?v=YYEJ_GUguHw [Accessed 24 October 2022].

Carleton, P., 2022. *What should I do if my bank accout is hacked.* [Online] Available at: https://www.finder.com/bank-account-is-hacked [Accessed 22 October 2022].

Cote, C., 2021. *Predictive analytics.* [Online] Available at: https://online.hbs.edu/blog/post/predictive-analytics [Accessed 22 April 2022].

Koehrsen, W., 2018. Overfitting vs. underfitting: A complete example. *Towards Data Science.*

LaValley, M., 2008. Logistic Regression. *Circulation,* 117(18), pp. 2395-2399.

Muller, A. & Guido, S., 2017. *Introduction to Machine Learning with Python.* 1st ed. United States of America: O'Reilly.

Olken, F. & Rotem, D., 1995. Random sampling from databases: a survey.. *Statistics and Computing 5,* pp. 25-42.

Pal, M., 2005. Random forest classifier for remote sensing classification.. *International journal of remote sensing,* 26(1), pp. 217-222.

Siramdasu, V., 2022. *Fraudulent Transaction Prediction.* [Online] Available at: https://www.kaggle.com/datasets/vardhansiramdasu/fraudulent-transactions-prediction [Accessed 22 October 2022].

Stefanski, R., 2022. *What is Data Wrangling: 6 Important Steps.* [Online] Available at: https://hevodata.com/learn/data-wrangling/ [Accessed 25 October 2022].

The Independent Institute of Education, 2022. *Data Analytics Module Manual ,* s.l.: s.n.